

Tentamen Programmeermethoden

Vrijdag 9 augustus 1996, 9.00 – 12.00 uur

Rijksuniversiteit Leiden—Informatica

Bij alle te schrijven functies moeten de variabelen in de heading voorkomen (niet stiekem globale variabelen gebruiken). De opgaven tellen alle vier even zwaar mee.

1. Gegeven zijn `const int n = 1024;` en `typedef int array[n];`. Een variabele `A` van type `array` bevat een rij onderling verschillende getallen. Bij deze opgave gaan we `A` uiteindelijk sorteren met behulp van *shellsort*.
 - a. Schrijf een C++-functie `wissel (x,y)` die de waardes van de `int`-argumenten `x` en `y` verwisselt.
 - b. Schrijf een C++-functie `hsorteer (A,h)` die ervoor zorgt dat alle array-elementen die op afstand `h` (een deler van `n`) van elkaar staan ten opzichte van elkaar gesorteerd worden. Dus na afloop moet gelden dat `A[i] < A[i+h]` voor alle `i` waarvoor `i` en `i+h` geldige array-indices zijn. Er mogen alleen vergelijkingen worden gedaan tussen array-elementen die op afstand `h` van elkaar liggen, dus “gewoon” sorteren is niet goed. Hint: doe bubblesort op de `n/h` getallen `A[k],A[k+h],A[k+2h],...` voor `k = 0,1,...,h-1`.
 - c. Schrijf een C++-functie `sorteer (A)` die het array `A` olopend sorteert door herhaald `hsorteer (A,h)` aan te roepen, en wel voor `h` achtereenvolgens `n/2,n/4,...,1`.
 - d. Hoeveel vergelijkingen tussen array-elementen doet het algoritme van onderdeel b ongeveer bij vaste `h`?
 - e. (bonus-opgave) Hoeveel vergelijkingen tussen array-elementen doet het sorteer-algoritme van onderdeel c ongeveer? Gebruik:

$$1 + 1/2 + 1/4 + 1/8 + \dots + 1/2^t = 2 - 1/2^t.$$

2. a. Bij een functie kun je te maken hebben met call by value en call by reference, en ook met locale en globale variabelen. Verder heb je ook nog formele en actuele parameters. Leg deze zes begrippen aan de hand van een klein voorbeeld duidelijk uit.
- b. Gegeven zijn de volgende functies:

```
int holmes (int& r, int s) {
    r++; s--; return ( r + 2 ) * ( s - 2 ); } // holmes
void watson (int& p, int& q) {
    int h = 5; q = q + holmes (h,q) + p; p = 2 * p; k--;
    cout << "watson" << p << q << a << k << h << endl; } // watson
```

Neem aan dat de waarde van de *globale variabelen* `a` en `k` van type `int` bij binnenkomst van `watson` respectievelijk 4 en 11 zijn. Wat gebeurt er bij de aanroep `watson (a,k);`, gevolgd door `cout << a << k << endl;`? Wat wordt er afgedrukt? Probeer duidelijke uitleg te geven.

- c. Idem, maar nu zonder `&` in de twee headings.
- d. En wat gebeurt er als de aanroep `watson (a,a);` wordt, gevolgd door het statement `cout << a << k << endl;` (`a` en `k` van type `int` met waarde 7 respectievelijk 1; `&` er wel bij)?
- e. Als onderdeel d, maar nu met `int& s` in de heading van `holmes` in plaats van `int s`. Leg uit waarom er verschillende antwoorden mogelijk zijn.

3. Gegeven zijn:

```
const int m = 10; const int n = 40; typedef int rechthoek[m][n];
enum boolean {False = 0, True = 1};
```

Een variabele `R` van type `rechthoek` bevat gehele getallen, groter dan of gelijk aan 0.

a. Schrijf een integer C++-functie `maxi (R,i,j)` die de grootste waarde van de vier directe burens van het vakje uit de `i`-de rij, `j`-de kolom, van de `rechthoek R` teruggeeft. De directe burens zijn de vakjes die onmiddellijk horizontaal of verticaal aan het gegeven vakje grenzen; randvakjes hebben uiteraard twee of drie directe burens.

b. Schrijf een integer C++-functie `locmax (R)` die het aantal *locale maxima* uit de `rechthoek R` oplevert. Een lokaal maximum is een array-element dat groter dan of gelijk is aan zijn directe burens.

c. Schrijf een boolean C++-functie `pad (R,a,b,i,j)` die `True` oplevert precies dan als er een *strikt stijgend pad* van het vakje in rij `a`, kolom `b`, naar het vakje in rij `i`, kolom `j`, is. Een strikt stijgend pad is het volgende: ga vanuit een vakje steeds naar het hoogste directe buurvakje, dat ook nog eens hoger moet zijn dan het oorspronkelijke. Als zo'n buurvakje niet bestaat, stopt het pad. Gebruik een aangepaste versie van de functie van onderdeel a, die ook nog rij- en kolomindex van de hoogste buur als parameter oplevert.

4. Gegeven zijn de volgende types:

```
class vakje { int info;          // een getal
              vakje* rechts;    // wijst naar vakje er rechts naast
              vakje* onder;     // wijst naar vakje er onder
}; // vakje
typedef vakje* wijzer;
```

Met behulp hiervan kan een lijst worden opgebouwd. Neem aan dat als het `info`-veld een waarde groter dan 0 bevat, de pointer `onder` `NULL` is, terwijl de pointer `rechts` `NULL` is als het `info`-veld een waarde kleiner dan 0 bevat. De andere pointer wijst dan naar het “volgende” vakje. Als de waarde 0 is, zijn beide pointers `NULL`. Een voorbeeld, met `ingang` van type `wijzer` (het vakje met -2 erin bevat hier een pointer (de “lijn” naar beneden) naar het vakje met 5 erin en een `NULL`-pointer):

```
ingang --> 3 --> -2
           |
           5 --> 10 --> 0
```

a. Schrijf een C++-functie `voegtoe (ingang,getal)` die de `int getal` in een vakje vooraan de niet-lege lijst `ingang` toevoegt. Denk eraan dat de werking afhangt van het al dan niet positief zijn van `getal`; neem aan dat `getal` niet 0 is.

b. Schrijf een C++-functie `klapom (ingang)` die de waarde van het `info`-veld uit het eerste vakje van de niet-lege lijst `ingang` door zijn tegengestelde vervangt, en de pointers op de juiste wijze aanpast. In bovenstaand voorbeeld zou 3 vervangen worden door -3, en zou het vakje met -2 erin er onder komen te hangen.

c. Schrijf een C++-functie `verwijder (ingang)` die het eerste vakje uit de lijst `ingang` verwijdert. Als de lijst leeg is hoeft er niets te worden gedaan.

d. In de functies bij a, b en c staat in de heading steeds de parameter `ingang`. Deze heb je al dan niet “by reference” gedeclareerd (met een `&`). Maakt het voor de werking van deze functies verschil uit of die `&` erbij staat? Leg ook uit wat er bij deze twee mogelijkheden precies gebeurt tijdens executie van de betreffende functie.

e. Schrijf een integer C++-functie `som (ingang)` die de som oplevert van alle in de lijst `ingang` voorkomende `info`-velden. Als de lijst leeg is moet er 0 uitkomen.